



Trabalhando com GitHub

Overview sobre Versionamento

Definimos Versionamento como o gerenciamento (armazenamento e recuperação) de versões de objetos, ao invés de apenas gerenciar os próprios objetos. O Versionamento também pode ser usado como um termo para o ato do usuário criar várias versões do mesmo objeto.

A definição acima exige definirmos o conceito mais fundamental de uma versão. Dar uma definição precisa disso não é fácil, pois o significado real depende muito do contexto e do modelo de versão.

Informalmente, notamos que "uma versão" não pode existir por si só, mas deve ser entendida como sendo uma versão de algo. Assim, podemos tentar a seguinte definição: Uma versão é uma instância concreta potencial de algum objeto (específico).

Dias (2016) aponta que as perguntas a serem realizadas para analisar a necessidade de um software para controle de versão, são quatro:

1. "Alguém já subscreveu o código de outra pessoa por acidente e acabou perdendo as alterações?"
2. "Têm dificuldades em saber quais as alterações efetuadas em um programa, quando foram feitas e quem fez?"
3. "Tem dificuldade em recuperar o código de uma versão anterior da que está em produção?"

4. “Têm problemas em manter variações do sistema ao mesmo tempo”

Caso alguma dessas perguntas acima tiveram um “sim” como resposta, logo se faz necessário um software para fazer o versionamento do código.

E seguindo essa linha de raciocínio podemos citar o git, que é um sistema open-source que serve para fazer o gerenciamento de versões. Ele foi desenvolvido por Linus Torvards, o mesmo desenvolvedor do Linux, que foi criado exatamente para gerenciar o desenvolvimento dos códigos do Linux, isso em 2005.

Segundo Buis (2018) “é o sistema de controle de versão mais usado por aí e sua influência é difícil de exagerar”.

Também não podemos deixar de mencionar que o versionador Git é um sistema para controle de versão “distribuído”, ou seja, ele não depende de um servidor centralizado.

Uma coisa muito interessante é que o git pode ser usado para controlar versões de vários formatos, como por exemplo, de códigos fonte, projetos de análise de dados, manuscritos, websites, apresentações, etc...

E então vem a pergunta, mas porque usar o Git? Existem várias razões para isso dentre elas temos:

- Ele é rápido;
- Não é necessário que tenhamos acesso direto ao servidor, ter uma conta no git já é o suficiente;

- Muito indicado para fazer o gerenciamento e a unificação simultânea do mesmo arquivo;
- E atualmente se tornou o principal protocolo de gerenciamento de versões.

Outra ferramenta que podemos destacar é o GitHub. Então podemos dizer que é uma rede social para gerenciar códigos e projetos. Segundo Marques (2019) "Se o Git é o coração do GitHub, então o Hub é a alma. O hub de GitHub é o que torna uma linha de comando, como o Git, a maior rede social para desenvolvedores do mundo."

Com isso podemos chamar o GitHub de "rede social", pois dentro dele é possível uma socialização entre vários usuários e é um portfólio para indicar o que estão desenvolvendo. Todavia, vale ressaltar algo muito importante, o GitHub não é apenas para desenvolvedores, ainda segundo Marques (2019): O GitHub é uma ótima plataforma que mudou o método de trabalho de desenvolvedores. Mas qualquer pessoa que deseja gerenciar seu projeto com eficiência e trabalhar com outros colaboradores também pode usar o GitHub.

GitHub é um serviço popular de compartilhamento de código social baseado na Web que utiliza o sistema de controle de versão distribuído Git. Tem se tornado uma ferramenta essencial em áreas de tecnologia que requerem colaboração, como desenvolvimento de software e redação técnica. Também está vendo uma adoção generalizada em outras áreas, transformando a forma como as pessoas colaboram em um repositório compartilhado. Um dos principais pontos fortes do GitHub está na conscientização e recursos de transparência que fornece aos membros da equipe, do

projeto e da comunidade. Essas características influenciam positivamente como pessoas contribuem para projetos.

Ferramentas para GitHub

As revisões de código são uma parte fundamental do ciclo de vida de desenvolvimento de software, permitindo que você identifique bugs antecipadamente. Se você realiza revisões de código em sua empresa, pode incorporar várias ferramentas para ter um fluxo de trabalho de desenvolvimento mais robusto e facilitar seu trabalho.

Vamos aqui indicar algumas ferramentas que podem ajudá-lo a melhorar seu fluxo de trabalho do GitHub de revisão de código. Alguns podem ser integrados diretamente no GitHub, enquanto outros funcionam como clientes de desktop autônomos. Vamos verificá-los.

Bolt (por WhiteSource)

O Bolt foi projetado para encontrar e ajudar a corrigir vulnerabilidades de código aberto, para fechar a lacuna entre desenvolvimento de código e segurança. Essa ferramenta é executada como um aplicativo no GitHub (bem como uma extensão no Azure DevOps). O Bolt funciona escaneando todos os seus repositórios e detectando componentes vulneráveis.

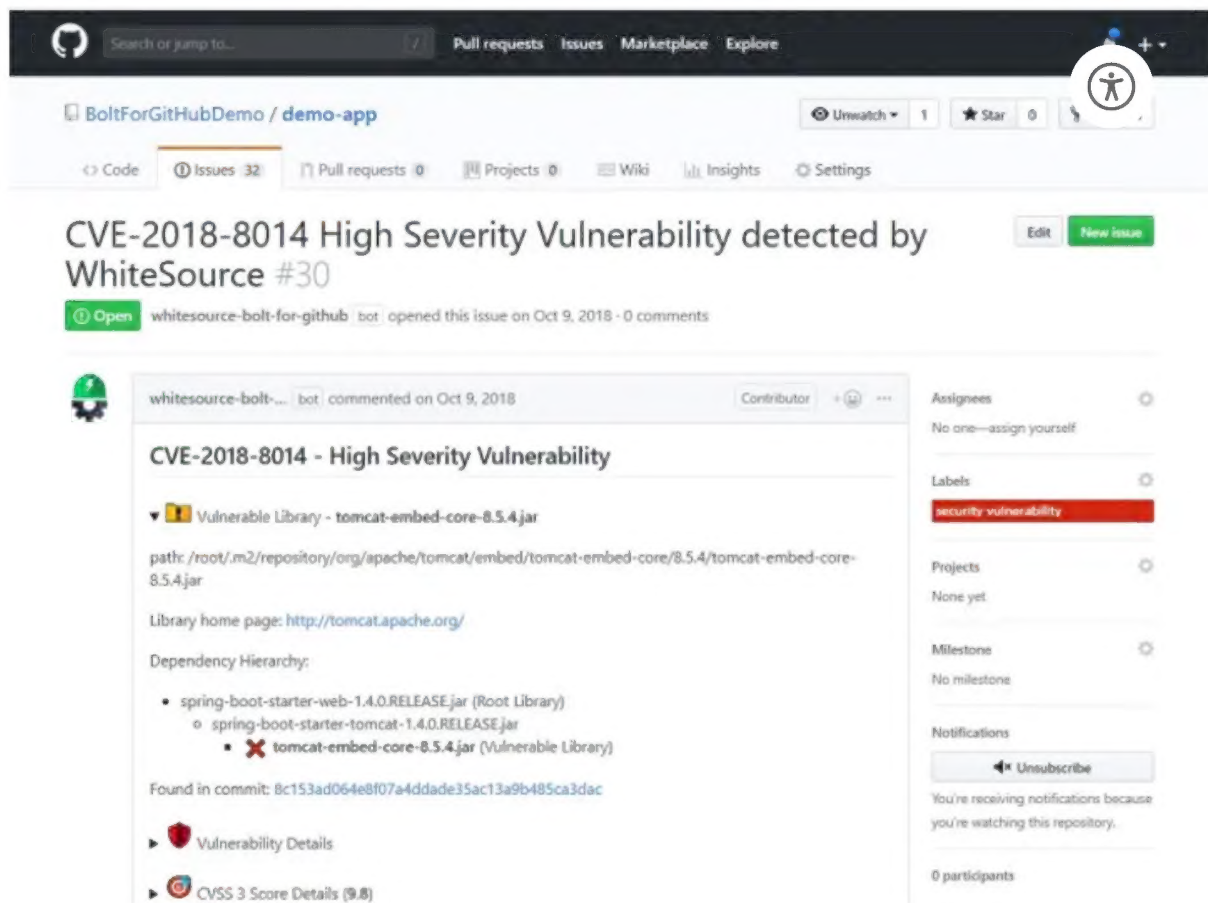


Figura 1 - Uma árvore de dependência de vulnerabilidade e uma correção sugerida com Bold, dentro do GitHub

DevHub

O DevHub é um cliente do GitHub focado em notificações, atividades e pull requests do GitHub. Com esta ferramenta, você está sempre atualizado com o que está acontecendo: você pode criar colunas para os repositórios e pessoas importantes para você e receber notificações push na área de trabalho sobre eles. O DevHub permite que você gerencie essas notificações e problemas, pull requests e atividades e marque itens para mais tarde. Você também pode aplicar filtros avançados em cada coluna, como filtrar por rótulo, ocultar toda a atividade do bot, entre outros.

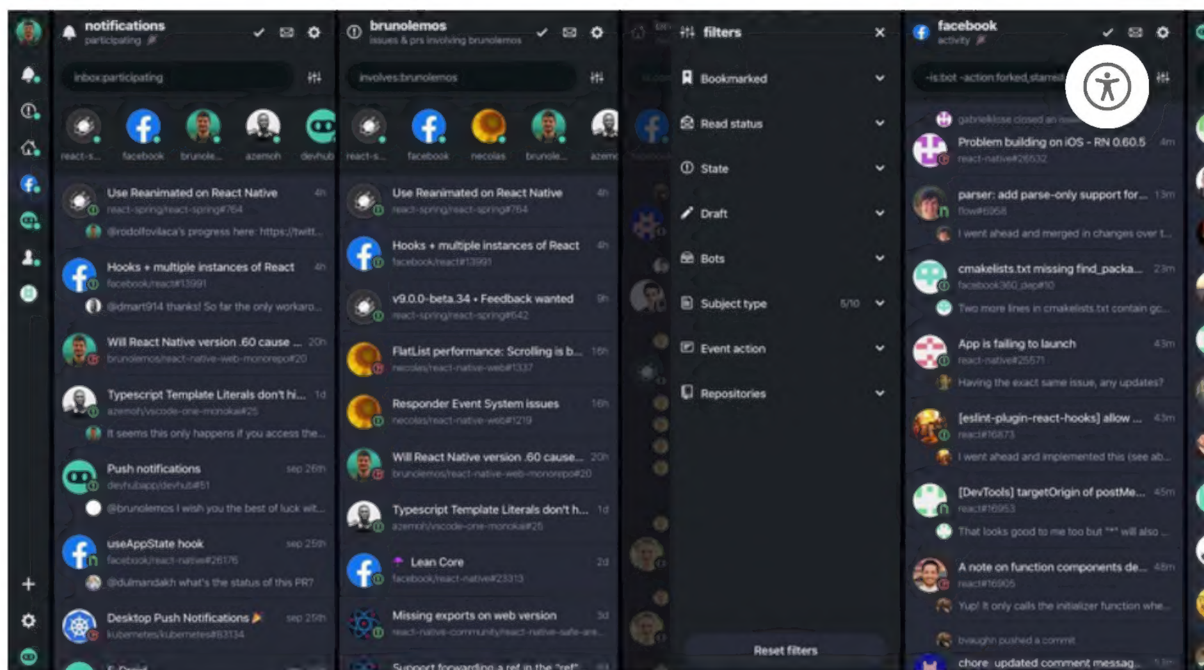


Figura 2: DevHub dashboard

Gitpod

O Gitpod é um serviço baseado em nuvem para configurar ambientes de desenvolvimento prontos para codificar. Ele permite que você crie instantaneamente e remotamente um ambiente de desenvolvedor diretamente de seu navegador ou IDE de desktop. Além disso, o Gitpod pode ser integrado ao GitHub (assim como ao GitLab e Bitbucket) e pré-compilar automaticamente ambientes de desenvolvimento para todas as suas ramificações.

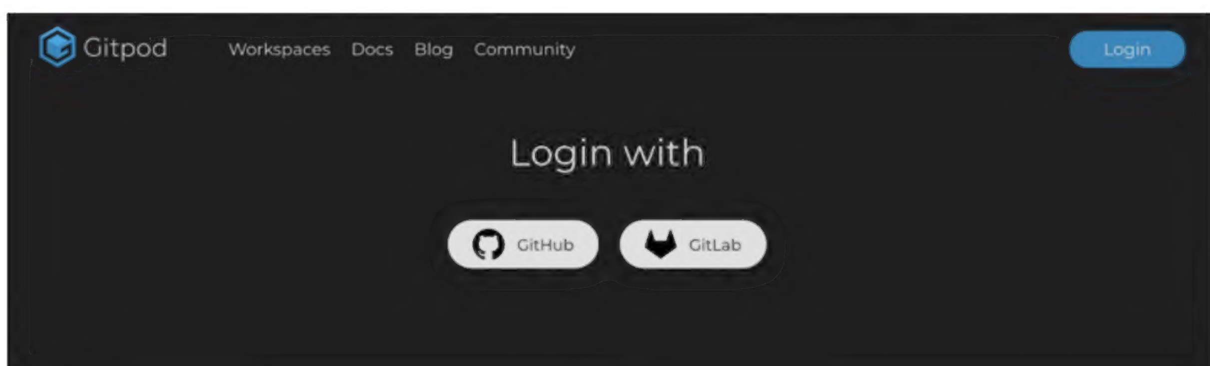


Figura 3: Login GitPod



Imgbot

O Imgbot é um robô amigável que otimiza suas imagens e economiza seu tempo, aproveitando o poder dos pull requests. Páginas da web com imagens otimizadas carregam mais rápido, e o Imgbot pode te ajudar nesse processo. Você só precisa instalá-lo em seus projetos do GitHub e parar de se preocupar em compactar suas imagens.

Search or jump to... Pull requests Issues Marketplace Explore

Code Issues 0 Pull requests 2 Actions Wiki Security Insights Settings

[ImgBot] Optimize images #127

Open imgbot wants to merge 1 commit into master from imgbot

Conversation 0 Commits 1 Checks 1 Files changed 10 +2 −2

imgbot bot commented 2 days ago

Beep boop. Your images are optimized!

Your image file size has been reduced by **50%**

▼ Details

File	Before	After	Percent reduction
/marketplace@3x-100.jpg	517.69kb	115.38kb	77.71%
/shirt.png	364.59kb	269.35kb	26.12%
/featured-marketplace.png	163.11kb	133.24kb	18.31%
/assets/svg/logo1.svg	11.70kb	10.01kb	14.45%
/logo.svg	11.70kb	10.01kb	14.44%
/wave.png	32.99kb	29.52kb	10.53%
Total :	**4,846.21kb**	**2,405.13kb**	**50.37%**

docs | repo | issues | swag | marketplace

[ImgBot] Optimize images Verified ✓ d3a1913

Reviewers dabutvin

Assignees dabutvin

Labels None yet

Projects None yet

Notifications Customize

Unsubscribe

You're receiving notifications because you're watching this repository.

2 participants

Lock conversation

Criação de Repositório

Antes de criar um repositório, definiremos o que é um repositório. Então falamos que repositório é um armazenamento virtual para projetos. Ele nos permite que salvemos versões do código, as quais poderá acessar no momento que precisar.

Os desenvolvedores podem seguir estas 10 etapas para criar um repositório GitHub:

1- Faça login no console administrativo do GitHub;

2- Mover para a página de repositórios do GitHub;

3- Clique no botão verde "Novo";

4- Isso abrirá o assistente de criação de repositório do GitHub;

5- Digite o nome do repositório GitHub;

6- Incluir uma descrição (opcional);

7- Escolha tornar este um repositório GitHub público ou privado;

8- Adicione um README (opcional);

9- Inclua um arquivo .gitignore para sua estrutura de desenvolvimento (opcional);

10- Escolha uma licença de uso justo.

Clique no botão verde "Criar Repositório" para finalizar o processo.

Figura 5: Criando um novo repositório

Versionando Códigos Fontes

Antes de começar é preciso entender que quando falamos em salvar e/ou compartilhar o código que está sendo desenvolvido dentro dos diretórios de rede, não é a mesma coisa que controlar a versão.

Isso é apenas uma forma de disponibilizar os arquivos de trabalho para os demais membros do time, porém abrindo o precedente para acidentes como a exclusão ou sobrescrita acidental de arquivos e o pior, sem controle do histórico das modificações.

A segunda forma é buscar informação sobre os modelos de versionamento disponíveis:

- Centralizado, onde em resumo trata-se de um modelo onde o versionamento é feito de forma centralizada através da conexão com um servidor.

- Descentralizado, onde neste caso cada diretório de trabalho será um repositório com histórico completo e também terá total disponibilidade para acompanhamento, revisões e não irá depender de acesso a um servidor central.

E por fim definir a ferramenta que vai permitir a utilização de qualquer um dos modelos escolhidos, ou quem sabe fazer uso de um modelo híbrido, mesclando as duas possibilidades.

Visto isso, vamos aos principais comandos para fazer o versionamento dos códigos.

git init

git branch -M master main

git remote add origin https://github.com/FaculdadeDescomplica/backend

git pull origin main

git remote -v

Sendo que:

Comando	Descrição
git init	Inicializa um repositório git local dentro da pasta projeto_integrador.
git branch -M master main	Renomeia a branch local master para main.
git remote add origin endereço_remoto	Associa o repositório local ao repositório remoto. O endereço_remoto será o endereço do seu repositório.
git pull origin main	Atualiza o seu repositório local com todos os arquivos disponíveis no repositório remoto.
git remote -v	Checa se o seu repositório local está conectado ao repositório remoto

Tabela 1: Explicação comandos git

Agora vamos seguir a sequência de comandos abaixo para para enviar o seu projeto para o repositório remoto backend:

git add .

git commit -m "Criar um projeto"

git push -u origin main

Podemos criar um *git clone* para criar um novo diretório dentro do local onde é executado e assim clonar o código da branch escolhida a partir de um repositório remoto para dentro dele. Além desta função ele pode mapear todas as branches existentes do repositório no momento do clone e para isso vamos escrever:

git branch –remotes

Um outro comando importante é o *git status* que será responsável por informar quais são os arquivos que sofreram alteração e precisam ser adicionados ao *commit*. Esses arquivos podem ou não estarem marcados como *staged*. Esses arquivos do tipo *staged* são arquivos que já foram selecionados pelo usuário que farão parte do próximo *commit*.

O *git log* irá mostrar o histórico de *commits* do *branch* atual. Acima de cada commit do log vamos encontrar um hash que funciona como se fosse um código identificador, além das informações relevantes de data, hora e autor do commit.

O comando git checkout tem por função primária e principal nos permitir navegar entre branches e commits. Com isso, você poderá acessar uma versão de código diferente em outra branch ou mesmo navegar para um commit de um ano atrás, por exemplo.

Para acessar uma branch diferente, funciona assim:

git checkout nome_da_branch

O git merge é o comando que nos permite mesclar o conteúdo de uma branch com o de uma segunda branch.

Quando acessamos nossa branch develop e podemos verificar que ela não possuía todos os arquivos necessários que estavam inclusos na branch master? Supondo que temos o interesse de atualizar a develop com o conteúdo mais recente da branch master, com isso podemos utilizar o comando git merge para fazer essa execução:

```
→  git-remote-example git:(master) ls -l
total 8
-rw-rw-r-- 1 raphael raphael  17 jun  4 22:31 example.txt
-rw-rw-r-- 1 raphael raphael 125 jun  6 19:29 README.md
-rw-rw-r-- 1 raphael raphael   0 jun  6 19:29 test.txt
→  git-remote-example git:(master) git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.
→  git-remote-example git:(develop) ls -l
total 4
-rw-rw-r-- 1 raphael raphael 17 jun  4 22:31 example.txt
→  git-remote-example git:(develop) git merge master
Updating 083e41b..ad94a51
Fast-forward
 README.md | 3 +++
 test.txt  | 0
 2 files changed, 3 insertions(+)
 create mode 100644 README.md
 create mode 100644 test.txt
→  git-remote-example git:(develop) ls -l
total 8
-rw-rw-r-- 1 raphael raphael  17 jun  4 22:31 example.txt
-rw-rw-r-- 1 raphael raphael 125 jun  6 19:29 README.md
-rw-rw-r-- 1 raphael raphael   0 jun  6 19:29 test.txt
```

Figura 6: git merge

Para finalizar, vamos mostrar o fluxo como um todo. Assim, considerando que todos os comandos descritos, a imagem abaixo, está ilustrando como eles se relacionam e também como funciona o fluxo de transferência de dados (no caso nosso código) dentro do nosso Git.

Fluxo de trabalho com Gitflow

Indiscutivelmente, uma das estratégias de ramificação do Git mais populares, o Git flow foi introduzido pelo desenvolvedor de software Vincent Driessen em 2010 com a intenção de simplificar o gerenciamento de versões.

Fundamentalmente, o fluxo do Git envolve isolar seu trabalho em diferentes tipos de branches.

No fluxo de trabalho do fluxo do Git, existem cinco tipos diferentes de branch:

- Principal
- Desenvolver
- Característica
- Liberar
- Correção

Git Flow: Ramo principal

A ramificação principal é comumente chamada de “mestre”; tomamos uma decisão intencional de evitar esse termo desatualizado e optamos por usar “principal”.

O objetivo da ramificação principal no fluxo de trabalho do Git é conter código pronto para produção que pode ser liberado.

No fluxo do Git, a ramificação principal é criada no início de um projeto e é mantida durante todo o processo de desenvolvimento. A ramificação pode ser marcada em vários commits para significar diferentes versões ou lançamentos do código, e outras ramificações serão mescladas na ramificação principal depois de terem sido suficientemente examinadas e testadas.

Git Flow: desenvolver branch

A ramificação de desenvolvimento é criada no início de um projeto e mantida durante todo o processo de desenvolvimento e contém código de pré-produção com recursos recém-desenvolvidos que estão em processo de teste.

Os recursos recém-criados devem ser baseados na ramificação de desenvolvimento e, em seguida, mesclados novamente quando estiverem prontos para teste.

Git Flow: ramificações de suporte

Ao desenvolver com o fluxo Git, existem três tipos de branches de suporte com diferentes propósitos: recurso, lançamento e hotfix.

Git Flow: ramificação de recurso

A ramificação de recurso é o tipo mais comum de ramificação no fluxo de trabalho do fluxo do Git. Ele é usado ao adicionar novos recursos ao seu código.

Ao trabalhar em um novo recurso, você iniciará um branch de recurso fora do branch de desenvolvimento e, em seguida, mesclar suas alterações de volta no branch de desenvolvimento quando o recurso for concluído e revisado adequadamente.

Git Flow: ramificação de lançamento

A ramificação de lançamento deve ser usada ao preparar novas versões de produção. Normalmente, o trabalho realizado nas ramificações de lançamento envolve retoques finais e pequenos bugs específicos para liberar novo código, com código que deve ser tratado separadamente da ramificação de desenvolvimento principal.

Git Flow: ramificação de hotfix

No fluxo do Git, a ramificação de hotfix é usada para dar rapidamente as alterações necessárias em sua ramificação principal.

A base da ramificação de hotfix deve ser sua ramificação principal e deve ser mesclada de volta nas ramificações principal e de desenvolvimento. Mesclar as alterações da ramificação de hotfix de volta na ramificação de desenvolvimento é fundamental para garantir que a correção persista na próxima vez que a ramificação principal for lançada.



Figura 7: O diagrama de fluxo do Git

Criação de Releases

As releases no GitHub são uma solução completa do GitHub para fornecer pacotes de software em arquivos binários junto com suas

notas de versão para cada versão do software. Os arquivos binários são uma ótima maneira de fornecer ao usuário uma ve do software na forma de código até um determinado ponto. Portanto, se você precisar do arquivo binário de um software XYZ versão 2.5, que está atualmente na versão 3.1, poderá obtê-lo rapidamente pelo GitHub. Além do código, as notas de versão do software também estão lá. Que, por sua vez, incluem detalhes da adição de novos recursos ou outras melhorias. Portanto, se você quiser saber sobre o software sem realmente instalá-lo, leia estas notas. Além disso, o recurso de lançamento ajuda pessoas de todo o mundo a ver como o software cresceu com o tempo e a usar seu arquivo binário também.

Criando uma versão do GitHub

O GitHub dá controle total ao desenvolvedor sobre os lançamentos. Você pode criar versões no GitHub de duas maneiras:

- Primeiro, através das tags já criadas.
- Segundo, criando uma versão nova/fresca.

Como criar releases no GitHub a partir de tags?

> Para visualizar as tags, abra a lista de tags no GitHub (Refer Tags In GitHub). Além disso, a opção de criar um release estará disponível à direita do nome da tag.

Pressione o botão destacado que diz "Criar versão" para ir para a próxima tela. Depois disso, você notará duas mudanças aqui.

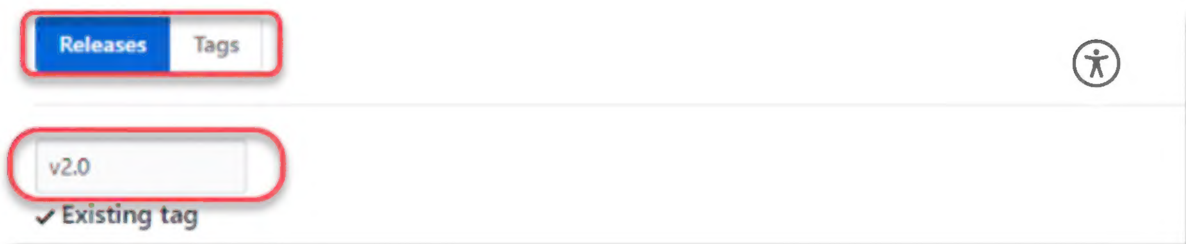


Figura 8. Releases

Primeiro, a guia mudou de “Tags” para “Releases”, indicando que agora estamos trabalhando com uma release.

Segundo, o nome da tag com a mensagem “Tag existente” menciona que estamos criando uma release a partir de uma tag existente.

Abaixo disso, teremos algumas opções para preencher para o nosso release. Podemos preenchê-los rapidamente:

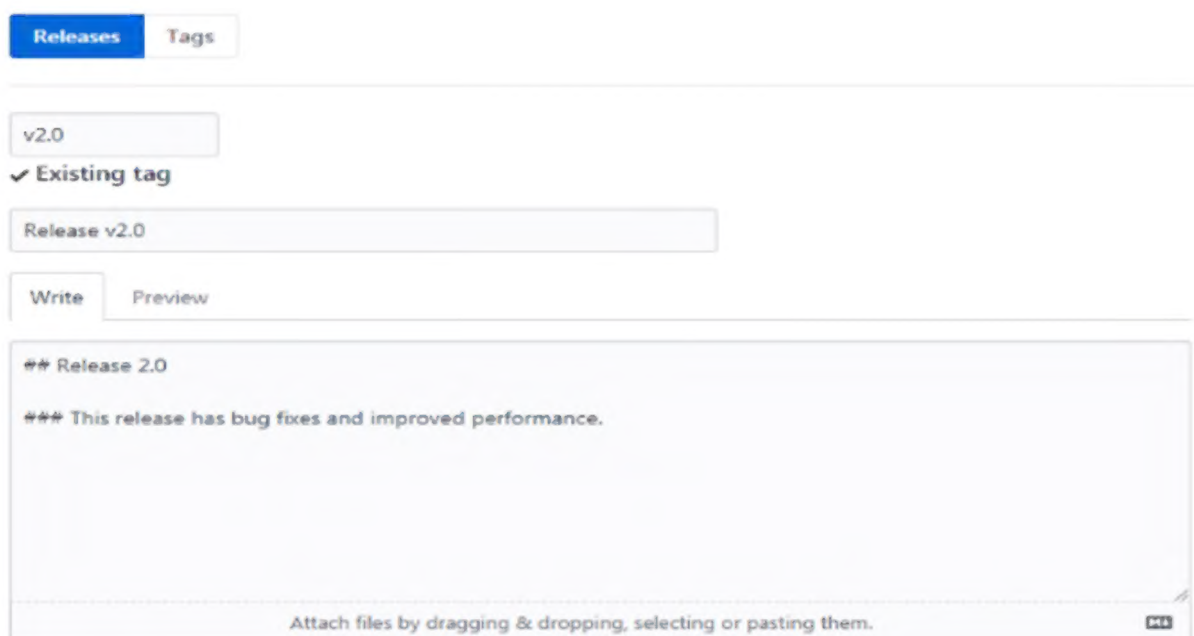


Figura 9. Preenchimento dos Releases



Deve-se notar que o campo de descrição suporta o formato markdown. Seria familiar para você se você já trabalhou com notebooks Jupyter. Mas você também pode aprender sobre o formato de remarcação. Em resumo, trata-se de estilizar a saída resultante por meio de notações específicas que aplicamos ao escrever o texto. Por exemplo, # significa H1 ou título de primeiro nível. Da mesma forma, ## significa um título de segundo nível e assim por diante. Preencha os campos indicados em conformidade.

A caixa de seleção aparecerá, perguntando se é um pré-release ou não. Durante este tempo, deixarei como está para denotar que a publicação deste release já aconteceu.

Figura10: Publish release

Pressione "Publish release" para publicar a liberação da tag existente.

Assim que você publicar o release no GitHub, podemos vê-lo na guia de release, que anteriormente mostrava apenas os nomes das tags.

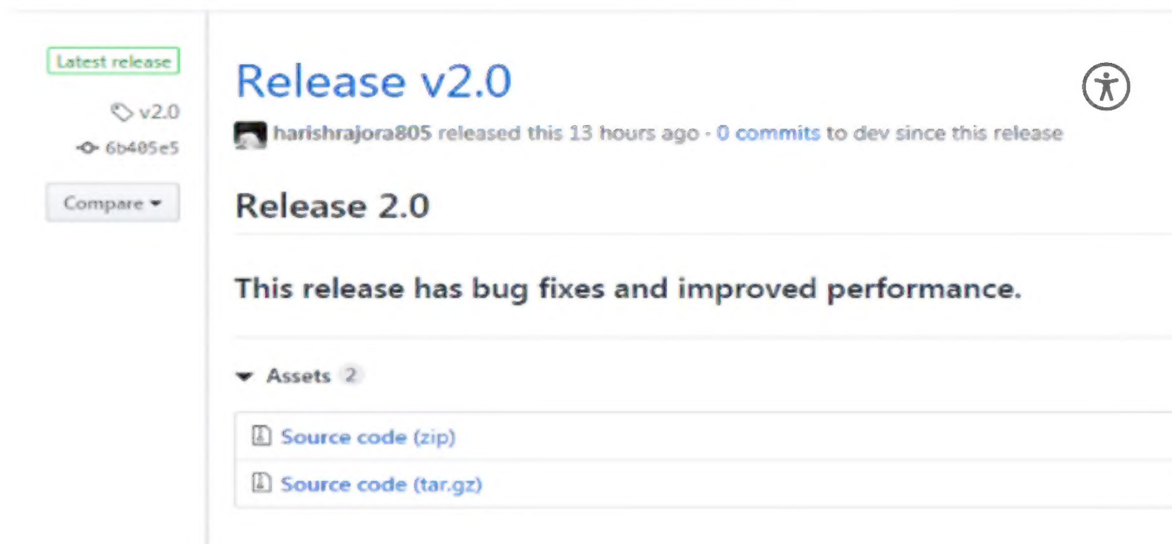


Figura 11: Guia de release

Atividade Extra

Para se aprofundar no assunto desta aula assista o vídeo Git - git flow na prática. A vídeo aula ensina na prática como criar um fluxo de Git Flow.

Canal: **Canal Angelo Luz**

Título a ser buscado no youtube: **Git - git flow na prática**

Referência Bibliográfica

BUIS, JUAN. **The impact of Git on software development**. Codacy Blog.Disponível em:<https://blog.codacy.com/the-impact-of-git-on-

software-development/>. (Acesso em 20 de junho de 2022).



DIAS, ANDRÉ. **Conceitos Básicos de Controle de Versão de Software** —Centralizado e D. Blog Pronus. Disponível em: <https://blog.pronus.io/posts/conceitos-basicos-de-controle-de-versao-de-software-centralizado-e-distribuido/>. (Acesso em 20 de junho de 2022)

MARQUES, BRENDON. **O que é GitHub e para que é usado? Hostinger Tutoriais**. Disponível em: <https://www.hostinger.com.br/tutoriais/o-que-github/>. (Acesso em 20 de junho de 2022)

STOREY, M.-A., Singer, L., Cleary, B., Figueira Filho, F., and Zagalsky, **A. The ® evolution of social media in software engineering. In Proceedings of the on Future of Software Engineering,** ACM (2014), 100–116.

Ir para exercício